|  |  |
| --- | --- |
| **Total Marks:** | **04** |
| **Obtained Marks:** |  |

**OBJECT ORIENTED PROGRAMING**

**Assignment no 4**

**Submitted to: SIR USMAN ALI**

**Student Name: Muhammad Hammas & Abdul samad**

**Reg Number: 2380176 & 2380147**

**\_\_\_\_**

***Instructions****: Copied or shown assignments will be marked zero. Late submissions are not entertained in any case.*

**Code:**

#include <iostream>

#include <fstream>

#include <string>

using namespace std;

class Account {

protected:

string name;

int accountNum;

double balance;

public:

Account(string n = "", int ac = 0, double bal = 0.0) : name(n), accountNum(ac), balance(bal) {}

string getName() {

return name;

}

int getAccountNum() {

return accountNum;

}

double getBalance() const {

return balance;

}

virtual void deposit(double amount) {

balance += amount;

}

virtual void withdraw(double amount) {

if (balance >= amount) {

balance -= amount;

cout << "\t\tWithdraw Successfully..." << endl;

} else {

cout << "\t\tInsufficient Balance...." << endl;

}

}

friend ofstream& operator<<(ofstream& ofs, const Account& acc);

friend ifstream& operator>>(ifstream& ifs, Account& acc);

};

class SavingsAccount : public Account {

public:

SavingsAccount(string n = "", int ac = 0, double bal = 0.0) : Account(n, ac, bal) {}

};

class CurrentAccount : public Account {

public:

CurrentAccount(string n = "", int ac = 0, double bal = 0.0) : Account(n, ac, bal) {}

};

class BankManagement {

private:

static const int MAX\_ACCOUNTS = 100; // Maximum number of accounts

Account\* accounts[MAX\_ACCOUNTS];

int numAccounts;

public:

BankManagement() : numAccounts(0) {

loadAccounts();

}

~BankManagement() {

saveAccounts();

for (int i = 0; i < numAccounts; i++) {

delete accounts[i];

}

}

void AddAccount(Account\* account) {

if (numAccounts < MAX\_ACCOUNTS) {

accounts[numAccounts++] = account;

} else {

cout << "\t\tMaximum number of accounts reached!" << endl;

}

}

void showAllAccounts() const {

cout << "\t\tAll Account Holders " << endl;

for (int i = 0; i < numAccounts; i++) {

cout << "Name: " << accounts[i]->getName() << " Account Number: " << accounts[i]->getAccountNum() << " Balance: " << accounts[i]->getBalance() << endl;

}

}

Account\* findAccount(int accountNum) const {

for (int i = 0; i < numAccounts; i++) {

if (accounts[i]->getAccountNum() == accountNum) {

return accounts[i];

}

}

return new Account("", -1, 0.0);

}

void saveAccounts() const {

ofstream ofs("accounts.txt");

if (!ofs) {

cout << "Error opening file for writing!" << endl;

return;

}

ofs << numAccounts << endl;

for (int i = 0; i < numAccounts; i++) {

}

}

void loadAccounts() {

ifstream ifs("accounts.txt");

if (!ifs) {

cout << "Error opening file for reading or file not found!" << endl;

return;

}

ifs >> numAccounts;

for (int i = 0; i < numAccounts; i++) {

string accountType;

ifs >> accountType;

Account\* account;

if (accountType == "class SavingsAccount") {

account = new SavingsAccount();

} else if (accountType == "class CurrentAccount") {

account = new CurrentAccount();

} else {

continue;

}

ifs >> \*account;

accounts[i] = account;

}

}

};

ofstream& operator<<(ofstream& ofs, const Account& acc) {

ofs << acc.name << endl;

ofs << acc.accountNum << endl;

ofs << acc.balance << endl;

return ofs;

}

ifstream& operator>>(ifstream& ifs, Account& acc) {

ifs >> acc.name;

ifs >> acc.accountNum;

ifs >> acc.balance;

return ifs;

}

int main() {

BankManagement bank;

int choice;

char op;

do {

cout << string(2, '\n');

cout << "\t\t::Bank Management System" << endl;

cout << "\t\t\tMain Menu" << endl;

cout << "\t\t1. Create New Savings Account" << endl;

cout << "\t\t2. Create New Current Account" << endl;

cout << "\t\t3. Show All Accounts" << endl;

cout << "\t\t4. Deposit Money" << endl;

cout << "\t\t5. Withdraw Money" << endl;

cout << "\t\t6. Exit" << endl;

cout << "\t\t-------------------------------" << endl;

cout << "\t\tEnter Your Choice :";

cin >> choice;

switch (choice) {

case 1: {

string name;

int accountNum;

double balance;

cout << "\t\tEnter Name :";

cin >> name;

cout << "\t\tEnter Account Number :";

cin >> accountNum;

cout << "\t\tEnter Initial Balance :";

cin >> balance;

bank.AddAccount(new SavingsAccount(name, accountNum, balance));

cout << "\t\tSavings Account Created Successfully...." << endl;

break;

}

case 2: {

string name;

int accountNum;

double balance;

cout << "\t\tEnter Name :";

cin >> name;

cout << "\t\tEnter Account Number :";

cin >> accountNum;

cout << "\t\tEnter Initial Balance :";

cin >> balance;

bank.AddAccount(new CurrentAccount(name, accountNum, balance));

cout << "\t\tCurrent Account Created Successfully...." << endl;

break;

}

case 3: {

bank.showAllAccounts();

break;

}

case 4: {

int accountNum;

double amount;

cout << "\t\tEnter Account Number to Deposit Money :";

cin >> accountNum;

Account\* account = bank.findAccount(accountNum);

if (account != new Account("", -1, 0.0)) {

cout << "\t\tEnter Amount to Deposit :";

cin >> amount;

account->deposit(amount);

cout << "\t\t" << amount << " Deposited Successfully ...." << endl;

} else {

cout << "\t\tAccount Not Found ..." << endl;

}

break;

}

case 5: {

int accountNum;

double amount;

cout << "\t\tEnter Account Number to Withdraw Money :";

cin >> accountNum;

Account\* account = bank.findAccount(accountNum);

if (account != new Account("", -1, 0.0)) {

cout << "\t\tEnter Amount to Withdraw :";

cin >> amount;

account->withdraw(amount);

} else {

cout << "\t\tAccount Not Found ..." << endl;

}

break;

}

case 6: {

return 0;

}

default: {

cout << "\t\tInvalid choice. Please try again." << endl;

break;

}

}

cout << "\t\tDo You Want to Continue or Exit [Yes=y/No=n] ??";

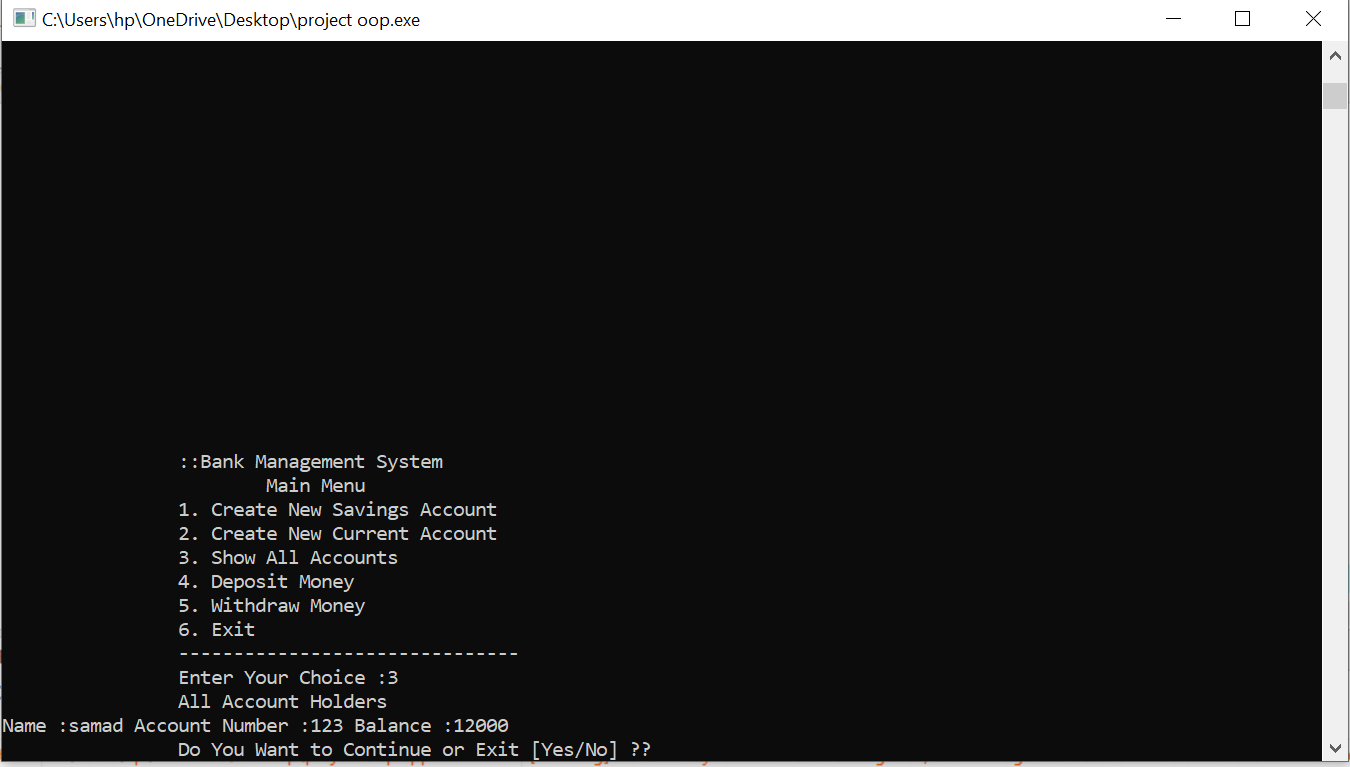
cin >> op;

} while (op == 'y' || op == 'Y');

return 0;

}

**Output:**

****